

# Solving algebraic word problems using text mining



By

**Tayyeba Rehman**

**NUST201463935MSEEC60014F**

Supervisor

**Dr. Sharifullah Khan**

**Department of Computing**

A thesis submitted in partial fulfillment of the requirements for the degree  
of Masters of Science in Information Technology (MS IT)

In

School of Electrical Engineering and Computer Science,  
National University of Sciences and Technology (NUST),  
Islamabad, Pakistan.

(March 2017)

# Approval

It is certified that the contents and form of the thesis entitled “**Solving algebraic word problems using text mining**” submitted by **Tayyeba Rehman** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Sharifullah Khan**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 1: **Dr. Sohail Iqbal**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 2: **Dr. Safdar Abbas**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 3: **Mr. Maajid Maqbool**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

# Abstract

Development of experimental techniques and computational models accelerated the pace of research and development in the field of mathematics and text mining. One of the major research area in the field of automation is solving word problems using Artificial Intelligence (AI) techniques. The existing research, for solving word problems using AI, is unable to provide on the spot solutions for a given word problem. The existing approaches, for solving algebraic and arithmetic word problems, have some limitations that need to be addressed. They are limited to give high accuracy for solving arithmetic problems only. They also require too much processing resources and time for extracting and simplifying features from the text data. In this research work, we proposed a template-based approach that has been developed by following a two-step process. The first step is to predict an equation template from a training dataset. The next step is to instantiate the predicted template with nouns and numbers. The features extraction in the proposed approach only relays on POS tags, NER and dependency relation; no need to extract all words from the given word problem. To validate the proposed methodology, a prototype system has been implemented. The system has been compared with the existing systems using their respective datasets and the proposed dataset. The experimental results show improvement in accuracy, with an average precision of 80.6% and average recall of 83.5%. In future, we intend to incorporate the co-reference resolution technique into our system to further improve its accuracy.

# Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECs) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECs or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Tayyeba Rehman**

Signature: \_\_\_\_\_

# Acknowledgment

Up and above everything all glory to **ALMIGHTY ALLAH**. The Beneficent, The most Merciful and Most Compassionate. It's a great blessing from Almighty Allah that He gave me the health and strength to do this research work.

This thesis would not have been possible without the inspiration and support of a number of wonderful individuals my thanks and appreciation to all of them for being part of this journey and making this thesis possible. I owe my deepest gratitude to my supervisors **Dr. Sharifullah Khan** for his patience, guidance, useful impact and availability throughout my research work. I could not have imagined having a better advisor and mentor for my thesis.

My special thanks to **Dr. Asad Anwar Butt, Dr. Sohail Iqbal, Dr. Safdar Abbas** and **Mr. Maajid Maqbool** for being my committee members. I gratefully acknowledge the contributions of **Dr. Azeem Abbass**.

**Tayyeba Rehman**

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Statement . . . . .	2
1.3	Research Goal . . . . .	3
1.4	Proposed System . . . . .	3
1.5	Thesis Outline . . . . .	4
<b>2</b>	<b>BACKGROUND</b>	<b>5</b>
2.1	Mathematical Background . . . . .	5
2.1.1	Basic Concepts . . . . .	6
2.1.2	Algebraic Equations . . . . .	8
2.1.2.1	Types of Equations . . . . .	8
2.1.2.2	Solving Methods for Linear System . . . . .	9
2.1.3	Word Problems . . . . .	10
2.2	Natural Language Processing . . . . .	11
2.2.1	Words and Tokenization . . . . .	12
2.2.2	Part of Speech Tagging (POST) . . . . .	13
2.2.3	Named Entity Recognition (NER) . . . . .	13
2.2.4	Dependency Parser . . . . .	15
2.2.5	Information Extraction (IE) . . . . .	16
2.2.6	Similarity Matrix . . . . .	17
2.3	Data Mining . . . . .	17
2.3.1	Classification Algorithms . . . . .	18
2.3.1.1	NaiveBayes . . . . .	18
2.3.1.2	SMO (Sequential Minimal Optimization) . . . . .	18
2.3.1.3	KStar . . . . .	19
2.3.1.4	RandomForest . . . . .	19
2.3.2	Text Mining . . . . .	20
2.4	Equation Solver . . . . .	21

<b>3</b>	<b>LITERATURE REVIEW</b>	<b>23</b>
3.1	Only NLP based approaches . . . . .	23
3.2	NLP and ML based approaches . . . . .	24
3.3	Critical Analysis . . . . .	26
<b>4</b>	<b>PROPOSED SYSTEM DESIGN</b>	<b>27</b>
4.1	Natural Language Processing (NLP) Module . . . . .	28
4.1.1	Spell Check . . . . .	30
4.1.2	Stemming . . . . .	31
4.1.3	Stop-word Remover . . . . .	32
4.1.4	POS Tagger (POST) . . . . .	32
4.1.5	Named Entity Recognition (NER) . . . . .	32
4.1.6	Dependency Parsing . . . . .	33
4.2	Equation Prediction Module . . . . .	34
4.2.1	Term Frequency and Similarity Matrix . . . . .	35
4.2.2	Classifier Algorithm . . . . .	37
4.3	Equation Solving Module . . . . .	38
<b>5</b>	<b>IMPLEMENTATION AND EVALUATION</b>	<b>40</b>
5.1	System Implementation . . . . .	40
5.2	Dataset Specifications . . . . .	43
5.2.1	Online Source . . . . .	43
5.2.2	Book Source . . . . .	43
5.3	Evaluation Metrics . . . . .	44
5.4	Performance Evaluation . . . . .	46
<b>6</b>	<b>CONCLUSION AND FUTURE DIRECTION</b>	<b>48</b>
6.1	Conclusion . . . . .	48
6.2	Contributions . . . . .	49
6.3	System Limitations & Future Work . . . . .	49

# List of Abbreviations

---

---

<b>Abbreviations</b>	<b>Descriptions</b>
NLP	Natural Language Processing
ML	Machine Learning
RF	RandomForest
KNN	K Nearest Neighbor
SVM	Support Vector Machine
SMO	Sequential Minimal Optimization
NER	Named Entity Recognition
POS	Part Of Speech
Parse tree	Dependency Parser
AI	Artificial Intelligence
JEP	Java Expression Parser
ARIS	Artificial Immune Recognition System

---



# List of Figures

1.1	Solution towards Single and Multiple variable's equations . . . .	2
2.1	Applications of pure and applied math [1] . . . . .	6
2.2	Sequential Minimal Optimization . . . . .	19
2.3	Working of Ensemble . . . . .	20
2.4	Process for text classification . . . . .	20
4.1	System Architecture . . . . .	28
4.2	Pre-processing Module . . . . .	29
4.3	Algorithm for Neural Network Dependency Parser [2] . . . . .	34
4.4	Equation Prediction Module . . . . .	35
4.5	Random Forest Classifier . . . . .	37
4.6	Equation Solving Process . . . . .	38
5.1	Training dataset implementation . . . . .	41
5.2	Training through RF classifier . . . . .	42
5.3	Predicting through RF classifier . . . . .	42
5.4	Web-Based Interface . . . . .	43
5.5	Sample of the Dataset . . . . .	44

# List of Tables

2.1	Difference between Words, Token and POST . . . . .	12
2.2	The tagset, sorted alphabetically according to categories . . . . .	14
2.3	Dependency parsing for example 2 . . . . .	15
4.1	Root identification from tokens . . . . .	31
4.2	POS tags for example 3 . . . . .	33
4.3	NER for example 3 . . . . .	33
4.4	Dependency relations for a sample sentence . . . . .	35
4.5	Similarity Matrix on TF . . . . .	37
5.1	System Specification . . . . .	40
5.2	Software Specification . . . . .	40
5.3	Confusion matrix [3] . . . . .	45
5.4	Evaluation metrics using proposed's dataset . . . . .	46
5.5	Evaluation metrics using Hossemi et al. [4]'s dataset . . . . .	46
5.6	Evaluation metrics using Kushman et al. [5]'s dataset . . . . .	46

# Chapter 1

## INTRODUCTION

This chapter introduces the research work that has been carried out in this thesis. It includes motivation and problem definition, followed by a discussion of objectives.

### 1.1 Motivation

Linear algebra is a core course in all engineering disciplines, almost one exercise in each chapter is specific for word problems. Similarly, teaching the algebra words problems starts from the early education, i.e., grade 3 (age: 8-10 year). To help students learn algebra, an automated way is required to solve these word problems. Previously, several techniques [6] solve these problems by a computational method, i.e., a solution or answer is provided to a student rather than providing the actual equation. Moreover, testing agencies, i.e., Graduate Record Examination (GRE) include a separate analytical section containing word problems. Existing techniques has low accuracy due to their large feature set [4] [5]. This motivated us to develop a model to automatically solve word problems on a training corpus of questions that are paired with equations and class labels, requires less processing resources and gives high accuracy. To achieve this, we utilized the standard template-based algorithm that requires predefined template schemas and often labeled data. Moreover, the model utilized the rule-based template instantiator to extract nouns and constants from a problem for the selected template.

## 1.2 Problem Statement

Several methods have been proposed in the recent years to solve mathematics word problems using machine learning algorithms [4][5]. However, these methods have some limitations that are discussed below:

- Existing methods are unable to evaluate an equation system with high accuracy [5].
- Generally, evaluating a value against a single variable in an equation is much easier than multiples. Therefore, the existing methods offer higher accuracy for arithmetic word problems in comparison of algebraic [4] [7], as arithmetic word problems revolve around one single variable. Figure 1.1 shows the solution towards a single and multiple variables' equations.

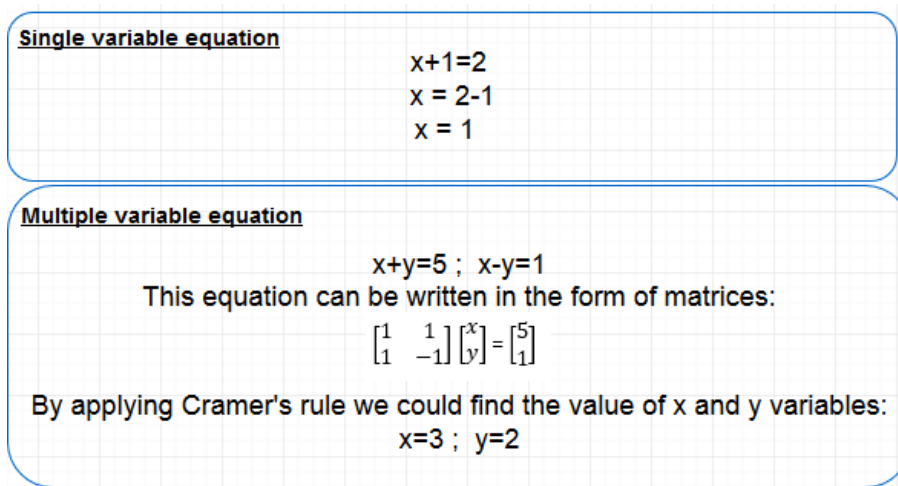


Figure 1.1: Solution towards Single and Multiple variable's equations

- They are implemented on 3<sup>rd</sup> and 4<sup>th</sup> grade's dataset, so only provide high accuracy toward 3<sup>rd</sup> and 4<sup>th</sup> grade students [4].
- As existing approaches, include lemmas of the problem text beside with Part Of Speech (POS) tags and Named Entity Recognition (NER), so it would take time to learn a model [8]. Time consumption has likewise relied upon the extent of the dataset and average length of a problem. In addition, existing models require supplementary time after learning a model from training and prediction, because instantiating variables and constants required nouns and numbers from a question.

However, training and predicting a model take much time as compared to instantiating [5].

- Additionally, existing approaches require high computing resources to learn a model for automatically solving algebraic word problems [5] [4].

### 1.3 Research Goal

The overall goal of the work, described in this thesis, is to build a highly accurate model for extracting equations from a given word problem. The proposed model will be able to drive and solve equations through Natural Language Processing (NLP) and classification techniques.

### 1.4 Proposed System

The present work solves the algebraic word problems using machine learning technique (i.e., Random Forest) by predicting a correct equation template. The feature set used in the proposed technique relies on the part-of-speech (POS) tags, named entity recognition (NER) and dependency type and relations. The proposed method has three main modules; (i) Natural Language Processing (NLP), (ii) Equation Prediction (iii) Equation Solving. In the first module, the documents are preprocessed to convert raw text into a well-organized sequence of linguistically-meaningful and machine understandable units. Additionally, NLP techniques are applied to extract useful features from a dataset. In equation prediction, module an equation template is predicted through classification. Finally, in the equation solving module, an equation template is instantiated and solved respectively. Precisely, the proposed system revolves around two steps that are:

1. A template selected through a classification algorithm that defines a structure for the equation system.
2. Equation template is instantiated with numbers and nouns extracted from the math word problem.

The accuracy of the proposed technique is higher than its predecessors due to the use of Random Forest classifier on the reduced features set. The proposed system is not only evaluated against the existing methodology but also has been evaluated against proposed's dataset (i.e., a dataset from two different sources), Hosseni et al. [4]'s dataset, and Kushman et al. [5]'s dataset. Experimental results demonstrate an improvement in accuracy with an average precision of 80.6%, recall of 83.5% and f-measure of 81.2%.

## 1.5 Thesis Outline

The rest of the document is organized as follows: Chapter 2 presents a background to some of the linear algebra terms used in this thesis and defines terminology from the domains of natural language processing and text mining. Chapter 3 discusses various related approaches, along with their critical analysis. In chapter 4, we give a detailed description of the proposed system methodology, explaining in detail the process of solving linear algebraic words problem containing up to two variables. Chapter 5 gives a complete overview of implementation details and describes the experimental results and a comparison with the existing systems. Concluding remarks and future work is presented in chapter 6.

# Chapter 2

## BACKGROUND

This chapter is going to explain some terms that are used throughout the thesis. Our study uses terminology from different domains such as mathematics, information extraction, natural language processing and text mining. So here we will concisely touch upon these domains, specifically the following fundamental areas:

1. Mathematical background
2. Natural Language Processing (NLP)
3. Data mining
4. Equation solver

Firstly, as the research problem is coming from mathematics, so it is important to have a mathematical background, with emphasis on basic concepts like set and matrix, algebraic expressions and their equations, and word problems. Next, how techniques from natural language processing can be applied and how these techniques are used to help in equation(s) extraction.

### 2.1 Mathematical Background

Mathematics is the mother of all sciences including natural sciences, engineering, medicine, and even the social sciences. It is the core analytical tool for quantitative science. Algorithmic (computational methods) are used to investigate the relationships among the measured quantitative properties of a system that could be mathematically analyzed. Mathematics and its laws are encoded in the human mind and brain, and thus are a property of

the physical mechanism of the brain [9].

There are two broad categories of mathematics world that are:

1. Applied mathematics
2. Pure mathematics

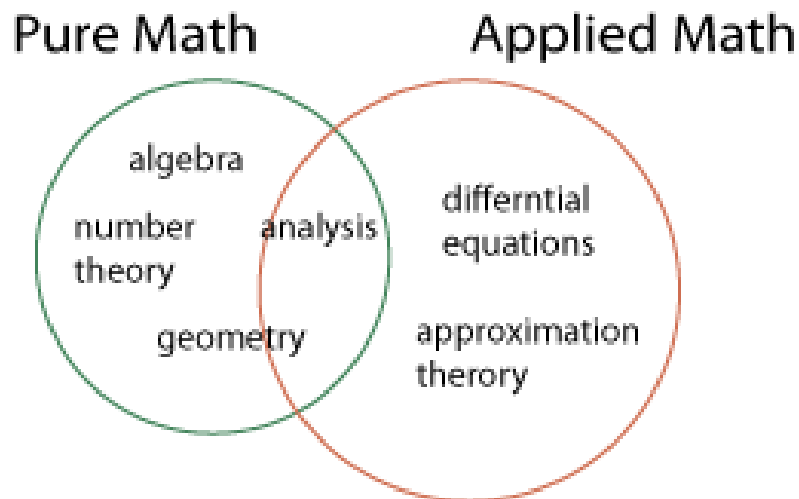


Figure 2.1: Applications of pure and applied math [1]

Pure mathematics is based on learning techniques of how to write proofs directly and try to generalize the pre-existing concepts [10]. While, applied mathematics have more focused on problem solving, modeling situations, where a mathematical model is a description of any system having mathematical concepts and language. Figure 2.1 shows the major applications of described categories of mathematics. Regarding this research area, we will focus on pure mathematics.

### 2.1.1 Basic Concepts

In this sub-section, we will discuss the basic concepts of pure mathematics as algebraic problems fall in this category. Algebra is a very powerful scientific device that is utilized to solve-out real-world problems in science, business, and numerous different fields. In this section, we begin with a review of fundamental definitions and documentations used to express algebraic terms such as: set, coefficients, algebraic expressions, and matrix. Also, we will further discussed the number theory, geometry, and algebra in this sub-section.



**Matrix** Matrices provide a theoretically and practically useful way of solving a linear system, where a linear system is a set of linear equations. Usually, there exist two techniques for solving linear system that are: Cramer's rule and Gaussian elimination. As there exist many libraries that are freely accessible for solving equations through Cramer's rule, we shall be using Cramer's rule to solve our equations. Before discussing Cramer's rule, it is worthy to define determinant for a matrix as it would be valuable while analyzing the matrix properties in a system of linear equations. The determinant for a matrix  $A$  can be denoted as  $\det(A)$  or  $|A|$ .

Let  $A$  be a  $2 \times 2$  matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Determinant of  $A$  is:

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

A linear system ( $AX = B$ ) only have a unique solution if and just if  $A$  is invertible (determinant for  $A$  not equal to zero). Cramer's formulas for two variables can be defined as:

$$x = \frac{|A_1|}{|A|}, |A_1| = \begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix} = b_1a_{22} - b_2a_{12} \quad (2.1)$$

$$y = \frac{|A_2|}{|A|}, |A_2| = \begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix} = b_2a_{11} - b_1a_{21} \quad (2.2)$$

As system of linear equations are precisely and concisely represented by a matrix, therefore to solve algebraic expressions researcher prefer to transform their set of linear equations into matrices.

**Number Theory** Arithmetic or Number theory is a sub-branch of pure mathematics that is used for the purpose of studying prime numbers as well as rational numbers. In other words, the number theory is the theory of the positive integers which states that each positive integer has a unique prime factorization [11]. Real numbers are divided into two categories: rational and irrational numbers. Rational numbers are the numbers that could be expressed in a form of  $\frac{p}{q}$  where  $p, q \in \mathbb{Z} \wedge q \neq 0$ . The numbers  $\sqrt{36}, 3, 9, 4$ , etc., are rational numbers. Whereas irrational numbers are the numbers that could not be expressed in a form of  $\frac{p}{q}$  where,  $q \in \mathbb{Z}$  and  $q \neq 0$ . The numbers  $\sqrt{5}, \sqrt{3}, \frac{7}{\sqrt{3}}, \frac{\sqrt{3}}{16}$  are irrational numbers.

Number theory is taught to computer researchers in their discrete mathematics course. Also numerical, image processing and cryptography are applications to it continuous theory.

**Algebra** The word algebra comes from the Arabic word “AL-JABR” that have a foundation from the al-Khwarizmi book titled *Ilm al-jabr wa'l-mubala*. Algebra is a subdivision of mathematics that helps the individual to evaluate the mathematical symbols through certain rules that are used for manipulating these symbols. Algebra differs from arithmetic in the context of using abstractions and generalization as algebraic problems either have unknown variable or a variable that allowed to take on many values. We will further discuss this area in the following subsection.

A variable could be represented through a symbol that would be used to represent an unknown number in an equation or expression. There exist two types of expressions: numerical expression (arithmetic expression) and algebraic expression. Numerical expression or arithmetic expression could be represented through two or more numbers that always gives a single answer. The expression  $7 + 8$  represents a single number that is 15, so it is a numerical expression. Whereas, the algebraic expression includes one or more variables. The expression  $7 + x$  represents a value that can be changed over time, i.e., if  $x$  is 1, then it has a value of 8. If  $x$  is 2, then the expression has a value of 9, so  $7 + x$  is an algebraic expression.

## 2.1.2 Algebraic Equations

An algebraic equation or algebraic expression is a combination of numbers, variables, and exponents. Typically, three types of equations in algebra: linear, quadratic, and polynomial equation. These types are further discussed in the following subsection.

### 2.1.2.1 Types of Equations

**Polynomial Equations** A polynomial in  $x$  is an expression of the form:  $a_n x^n + a_{(n-1)} x^{(n-1)} + \dots + a_1 x^1 + a_0$ ,  $a_n \neq 0$ . The highest power of  $x$  in a polynomial is called the degree of the polynomial. Remainder and Factor Theorem helps to solve a polynomial equation. Remainder Theorem for polynomial function tell us that the remainder obtained by dividing polynomial by  $x - a$  is same as the value of this polynomial function at  $x = a$ . While, factor theorem is used to check if the given linear polynomial is a factor of the given function or not, i.e., to determine if a given linear

polynomial  $x - a$  is a factor of polynomial function  $f(x)$ , all we need is to check whether  $f(a) = 0$ .

**Quadratic Equations** A quadratic equation or second degree polynomial equation in term of  $x$  is any equation that could be represented through a form  $ax^2 + bx + c = 0$  where  $a$ ,  $b$  and  $c$  are constants and  $a \neq 0$ . There are three basic techniques for solving these equations:

1. By factorization
2. By completing squares, extracting square roots.
3. By applying the quadric formula  $x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$

**Linear Equations** A polynomial equation of degree one is called linear equation. A three-variable linear equation can be represented in the form of:

$$ax + by + cz = k$$

These equations might be homogeneous or non-homogeneous. Any equation that could be written into the form of:

$$ax + by = k$$

where,  $a \neq 0, b \neq 0, k \neq 0$ , is called a non-homogeneous linear equation for two variables  $x$  and  $y$ . If in the above equation,  $k = 0$ , that is,  $ax + by = 0$ , then it would be a homogeneous linear equation in  $x$  and  $y$ .

The combination of more than one equations makes a system of linear equations that can be consistent or inconsistent. A system of linear equations is said to be inconsistent if the system does not have any solution and vice versa. The methods for solving *linear equations system* are described in the following subsection.

### 2.1.2.2 Solving Methods for Linear System

There exist three basic techniques for the solution of *linear equations system*, these are:

1. By substitution
2. By elimination
3. By using Cramer's formula

**By Substitution** Consider

$$ax + by = 0 \quad (2.3)$$

$$cx + dy = 0 \quad (2.4)$$

From equation 2.3  $x = -by/a$

By substitute the value of  $x$  into equation 2.4 will give us the exact solution for  $x$ .

**By Elimination** Consider  $ax + by = 0; cx + dy = 0$ . To eliminate one variable (suppose  $x$ ), first we will try to make same coefficient for at least one variable in each equation. So,

$\frac{1}{a}(ax + by = 0)$ , by multiplying both side by  $\frac{1}{a}$

$\frac{1}{c}(cx + dy = 0)$ , by multiplying both side by  $\frac{1}{c}$

Difference of the equation would eliminate  $x$  and give us the value for  $y$ . For example:

$$x + \frac{b}{a}y = 0 \quad (2.5)$$

$$x + \frac{d}{c}y = 0 \quad (2.6)$$

$$\begin{aligned} & \overline{\left(\frac{b}{a} - \frac{d}{c}\right)y = 0} \\ & y = \frac{1}{\left(\frac{b}{a} - \frac{d}{c}\right)} \end{aligned} \quad (2.7)$$

**By Cramer's Rule** System of linear equation can be solved through Cramer's formulas that can be defined as:

$x_n = \frac{|A_n|}{|A|}$ , where  $n$  define the number of variables and  $|A_n| = \begin{vmatrix} a_{1n} & b_1 \\ a_{2n} & b_2 \end{vmatrix}$  (in case of two variables).

### 2.1.3 Word Problems

In mathematics, the word problem is the representation of any mathematics problem in the form of text rather than in mathematical notation. Mostly, arithmetic, combinatory logic, lambda calculus, linear and quadratic algebra problem can be represented in the form text. To model semantic relationships for finding a system of equation through the given text requires reasoning across sentence boundaries. This reasoning may be challenging for a student

but problem understanding is trivial. Whereas, for artificial intelligent systems the understanding of a math-ward problem is challenging.

A system needed to be developed that would help the students to develop a model from the question which has all necessary information to solve it. Also by relating the subject to real-life situations will promote mathematical interest and understanding.

**Example 1** *If Zain has 7 dollars and he uses 3 dollars to buy something. How much does he have now?*

*Model:* The model for example 1 “Remaining dollar:  $7 - 3$ ”

The common types of word problems are distance problems, area problem, work problems, age problems, percentage problems, numbers problems and mixtures problems.

## 2.2 Natural Language Processing

The main objective of Natural Language Processing is to develop such a useful technique that would help systems to derive meaning from natural language(s). For NLP processing the main challenge is to transform the human understandable language into a format that is understandable for a machine. There are three elements associated with any language statement: syntax, format, and semantics. A syntax of a language is the structural rule that a statement should follow to make sense. While semantics deals with the meaning and interpretation of a sentence. NLP application area includes automatic translation (interpretation) between languages, dialogue system (that allow an individual to interact and communicate with the system) and information extraction (that transforms unstructured text into database representations). Basically, NLP involves to develop a model of natural language phenomena and to design an algorithm for implementation of these models.

CoreStanfordNLP is commonly used NLP tool which is available as open source and is platform independent. Stanford parser is for providing description toward grammatical relations of a sentence. Where a sentence must be understood and effectively used by an individual without linguistic expertise. The following are the major tasks in Stanford parser:

1. Words and tokenization
2. Part of Speech (POS) tagging
3. Named Entity Recognition (NER)

## 4. Dependency Parser (Parse tree)

In the remainder of this section, we will describe the major tasks of Natural Language Processing in order to achieve the goals of NLP. Information extraction and similarity matrix are also described that is helpful for getting preprocessed training data.

## 2.2.1 Words and Tokenization

The concept of a token is different from that of a word (Table 2.1). Tokens include punctuation characters, while words do not. Tokens are bordered by white spaces or line break. For instance, example 1 has twenty-one tokens but nineteen words. Words can be characterized in many ways. The

Table 2.1: Difference between Words, Token and POST

<b>Text</b>	<b>Tokenization</b>	<b>POST</b>
it's	it+'s	it_PP's_VBS
we're	we+'re	we_PP're_VBP
you're	you+'ll	you_PP'll_MD
wanna	wan+na	wan_VVG na_TO

field of linguistics defines words as “symbols for concepts”, where symbol is the string used to denote a concept- a real world object. Just as a word is the unit of spoken language, the notion of token represents the unit of written text. Tokenization is the procedure of isolating up a content into grammatical segments, for instance into token or sentential components. This segmentation of a sentence into tokens is a pre-processing step for other annotations such as part-of-speech tagging and lemmatization. POST is the process of tagging each token according to their part-of-speech tag. Where lemmatization is a form of annotation that is all about grouping together word forms that belong to the same inflectional paradigm and then assign this group to each paradigm that corresponding uninflected form, which is known as lemma. Lemmatization is related to the process of POST, as the POS tagging is essential for the process of lemmatization. Lemmatization is viewed as a very helpful sort of corpus annotation since it gives extra exploration alternatives.

A number of associations exist between lexical units of a language, which characterize how the semantics of two words relate to each other. WordNet is a highly structured lexical database that maintains common relations between nouns, verbs, and adjectives. The most common lexical relations

among words are defined as under:

- *Synonymy* is the relation that exists between two words as by interchanging one for the other does not alter the meaning of the context, e.g., the terms “prevent” and “avoid” are related by synonymy.
- *Homonymy* exists where one word can represent distinct concepts.
- *Antonyms* Two words with opposite meaning are said to be antonyms of each other, e.g., a bit vs too much.

### 2.2.2 Part of Speech Tagging (POST)

POS tagging is a process about marking a part-of-speech tag in corresponding with a particular word or phrase. Mostly, NLP uses the Brown corpus model for getting POS tags of a corpus (i.e., a collection of related documents). The Brown Corpus consist of one million words of standard American English that were compiled by W.N. Francis and H. Kucera, Brown University in 1961. Brown customized the combined tags for words such as won't (MD\*) and I'd (PPSS+HVD). If a tag has an asterisk appended means it has a negator after their tag.

As POS tagging provides additional information about the grammatical structure of data (Table 2.2), so it would improve the searchability of a corpus. The vital concern in Part-of-Speech (POS) tagging, which has been debated since the 1950s, is whether the POS tagging should be based on meaning or on syntactic distribution. According to the first view, POST should be based solely on word meaning so it should always be tagged as a verb. The second view says that POST should be determined by the syntactic distribution of the word. For example, when “benefit” is used as a noun phrase, it should be tagged as a noun in that context; it should be tagged as a verb if it is the root of a verb phrase. As syntactic distribution complies with the principles of modern linguistics theories so we have chosen syntactic distribution as the main criterion for our POS tagging.

### 2.2.3 Named Entity Recognition (NER)

NER is also a subtask of IE (Information Extraction) that is all about labeling words of a text into pre-defined categories through NE Recognizer tool. A Named Entity Recognizer is for classifying a sequences of words which are their respective entity names, for example, individual and organization names, or genetic factor and protein names. For categorizing entities NE

Table 2.2: The tagset, sorted alphabetically according to categories

<b>Category</b>	<b>Tag</b>
Adjective	JJ
Adjective, comparative	JJR
Adjective, superlative	JJS
Adverb	RB
Adverb, comparative	RBR
Adverb, superlative	RBS
Cardinal Number	CD
Conjunction, coordinating	CC
Conjunction, subordinating	IN
Determiner	DT
Noun, generic	N
Noun, plural	NNS
Noun, singular or mass	NN
Preposition	IN
Pronoun, personal	PP
Pronoun, possessive	PP\$
there, existential	EX
to, infinitive use	TO
Verb, base form	VB = verb be VIH= verb have
Verb, generic	V
Verb, gerund or present participle	VBG = verb be WIG = verb have
Verb, modal	MD
Past participle	VBN = verb be VHN = verb have
Wh-adverb	WRB
Wh-determiner	WDT



Recognizer utilize seven class model (Location, Person, Organization, Money, Percent, Date, and Time). This model can be prepared on various data set, such as, MUC 6 and MUC 7. Additionally, this model use distributional similarity features for providing performance gain at the cost of increasing the size and runtime of data sets.

General Architecture for Text Engineering (GATE) and Stanford CoreNLP are Java based IDEs including Conditional Random Field (CRF) named entity recognition tool. Both IDEs are accompanying by well-engineered feature extractors designed for Named Entity Recognition.

## 2.2.4 Dependency Parser

Dependency relations is a connection between pair of words or phrases that could be characterizing as a relationship between these pairs. A dependency parser analyzes the linguistic structure of a sentence by means of establishing connections between root words and words which alter those roots. For instance, in the sentence *Alia tossed football*, *Alia* is the subject who is doing the tossing, and *football* is the item being *tossed* [2].

This connection can be signified as:

*subj(toss, Alia)*

*obj(toss, football)*

Table 2.3 shows a step by step process followed by GATE and Stanford CoreNLP to produce dependency relation of an example 2.

**Example 2** *Economic news had little effect on financial markets.*

Table 2.3: Dependency parsing for example 2

Trans.	Stack	Buffer	Relation
1	root	Economic adj, news noun, had verb, little  adj, effect noun, on prep, financial adj, markets noun	[]
2	root, Economic	news noun, had verb, little  adj, effect noun, on prep, financial adj, markets noun	amod[news, Economic]
3	root, news	had verb, little adj, effect noun, on prep, financial adj, markets noun	amod[news, Economic]
4	root, had	little adj, effect noun, on prep, financial adj, markets noun	amod[news, Economic] nsubj[had, news]
5	root, had, little	effect noun, on prep, financial adj, markets noun	amod[news, Economic] nsubj[had, news] root[root, had]

6	root,had	effect noun, on prep, financial adj, markets noun	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little]
7	root,had, effect	on prep, financial adj, markets noun	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little] dobj[had,effect]
8	root,had,on	financial adj, markets noun	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little] dobj[had,effect] prep[effect,on]
9	root,had,on, financial	markets noun	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little] dobj[had,effect] prep[effect,on] amod[market, financial]
10	root,had,on, market	[]	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little] dobj[had,effect] prep[effect,on] amod[market, financial] pmod[on,market]
11	root,had	[]	amod[news,Economic] nsubj[had,news] root[root,had] amod[effect,little] dobj[had,effect] prep[effect,on] amod[market, financial] pmod[on,market]

### 2.2.5 Information Extraction (IE)

IE is a task of extracting organized data from the substance of large text collections. In IE, the evidence is investigated. IE is different from Information Retrieval (IR) as IR uses specific keywords or queries to pull documents from large text collections, such as the Web dataset. After pulling out specific queries the specific documents are being analyzed.

Getting the facts can be hard with traditional query engines as traditional query engines are designed for retrieving whole documents. IR provides documents with the relevant information somewhere, whereas IE returns structured information at a much more profound level than traditional IR. If a database is developed through IE and connected back to the documents, then it can give an important alternative search tool. Results may not be constantly exact, but rather they can be valuable if connected back to the original text. IE technology is deployed for many other applications such as: Text Mining, Decision Support, Opinion Mining, Question Answering, Rich information retrieval and exploration, Semantic Annotation [12].

### 2.2.6 Similarity Matrix

Similarity can be explained by using different measures, like cosine similarity, correlation, distance measure or comparison of local histograms. TF (Term Frequency) and TF/IDF (Inverse Document Frequency) are measures to create a vector (matrix) in text mining. TF can be defined as a frequency of term  $i$  in document  $j$ , whereas TF/IDF is a frequency of term  $i$  in respect with the number of documents containing that term  $i$ . Following formulas further distinguish the difference between TF and TF/IDF.

$$tf_{ij} = \frac{N_{ij}}{N_j}$$

$$tf/idf_{ij} = tf_{ij} * \log\left(\frac{N}{df_i}\right)$$

Where,

$N_{ij}$  = number of times a word  $i$  has occurred in a  $j$  document

$N_j$  = is the total number of words in a  $j$  document

$df_i$  = number of documents containing  $i$

$N$  = total number of documents

The most utilized model of similarity matrix for text and document representation is Vector Space Model (VSM). VSM was designed and developed by Xufei Wang et al [3]. This model represents the document through an arrangement of fragments where each row depicts distinct terms and each column depicts cohesive segments.

## 2.3 Data Mining

Data mining (DM) is a procedure that is for mining knowledge from data. Weka, Orange and R are tools that are commonly used for performing these

functions of DM. We used Weka (An open source API written in java) as it is a suite for performing almost all ML and DM tasks. As our work is related with classification so in this section we have only concentrated on this function of DM. Also, a brief overview of text mining approaches is presented at the end of this section.

### 2.3.1 Classification Algorithms

Classification is for predicting a target class for each instance in data by simply following two steps: modeling and testing. In modeling, a classification algorithm finds a relationship between values by estimating the predictors and target class. Classification models are tested by applying these relationships to a data set in which the class assignments are unknown. Particularly used algorithms that are given by Weka are:

- NaiveBayes
- SMO (Sequential Minimal Optimization)
- KStar
- RandomForest

#### 2.3.1.1 NaiveBayes

Nave Bayes is a statistical classifier based on Bayes-theorem which performs probabilistic prediction.

$$P(C_i | X) = \frac{P(X | C_i) * P(C_i)}{P(X)}$$

Where,  $P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$

It is easy to implement and also good accuracy obtained in the case of a small dataset. However, dependencies are not modeled i.e., dependencies exist among variables such as patients, symptoms, and diseases.

#### 2.3.1.2 SMO (Sequential Minimal Optimization)

SMO is the name used for Support Vector Machine (SVM) in Weka that is based on linear and nonlinear regression. SVM examines for the hyperplane with the largest margin, i.e., maximum marginal hyperplane (Figure 2.2). Maximize margin by maximizing  $2/||w||$ , where  $w$  is the total weight for each vector. Training can be slow but accuracy is highly outstanding for classification and numeric prediction.

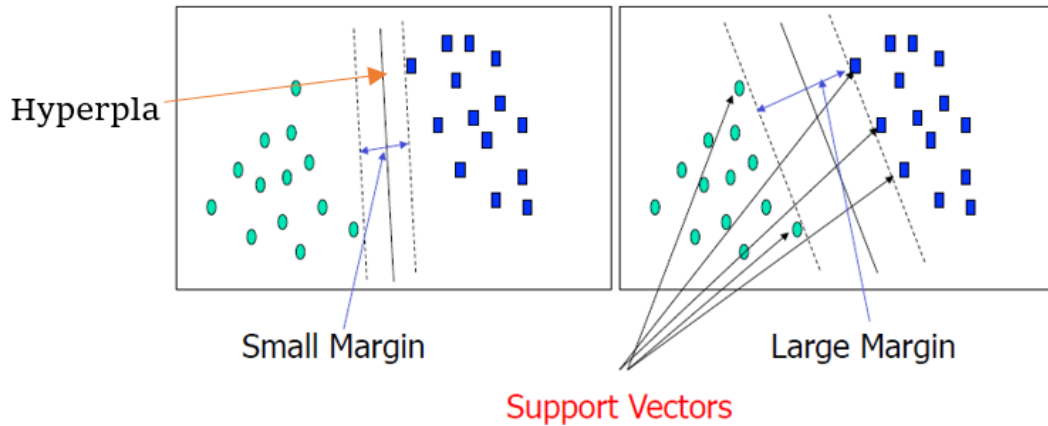


Figure 2.2: Sequential Minimal Optimization

### 2.3.1.3 KStar

KStar is an Instance-based Learning method known as KNN (K Nearest Neighbor), where classification is delayed till a new instance arrives. It classifies new instance based on some similarity functions such as Euclidean distance.

Consider instance:  $x = a_1(x), a_2(x), \dots, a_n(x)$

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

KNN calculate distance between new instance  $x_i$  with each of existing instance. The predicted class can be obtained either by voting or averaging of all k nearest neighbor classes. Learning process in KStar is very simple yet time consuming.

### 2.3.1.4 RandomForest

Random forest is an ensemble learning method for classification. Ensemble method used combination of models for getting high accuracy (Figure 2.3). Random forest used a series of k decision tree models and then chooses a model either having the most votes (Boosting) or averaging the prediction results of all models (Bagging). This algorithm gives more accurate result in the comparison of other algorithms described above but more robust to errors and outliers.

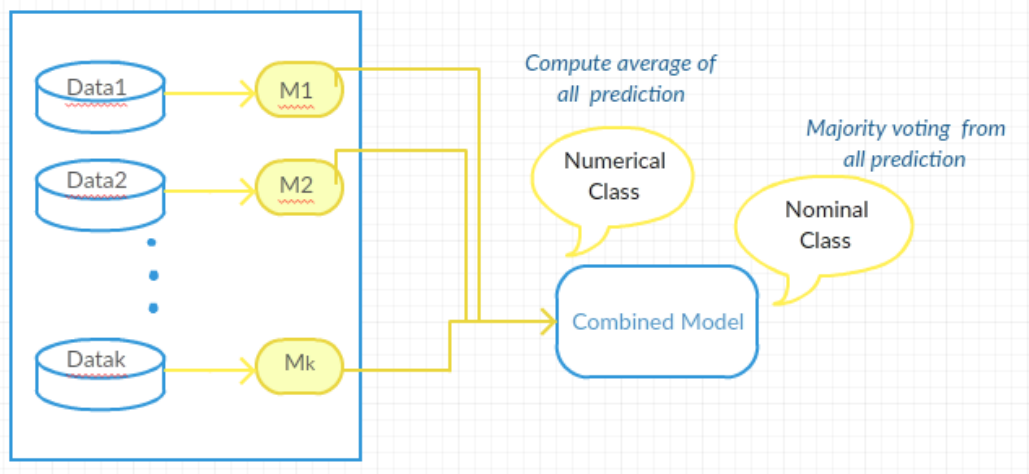


Figure 2.3: Working of Ensemble

### 2.3.2 Text Mining

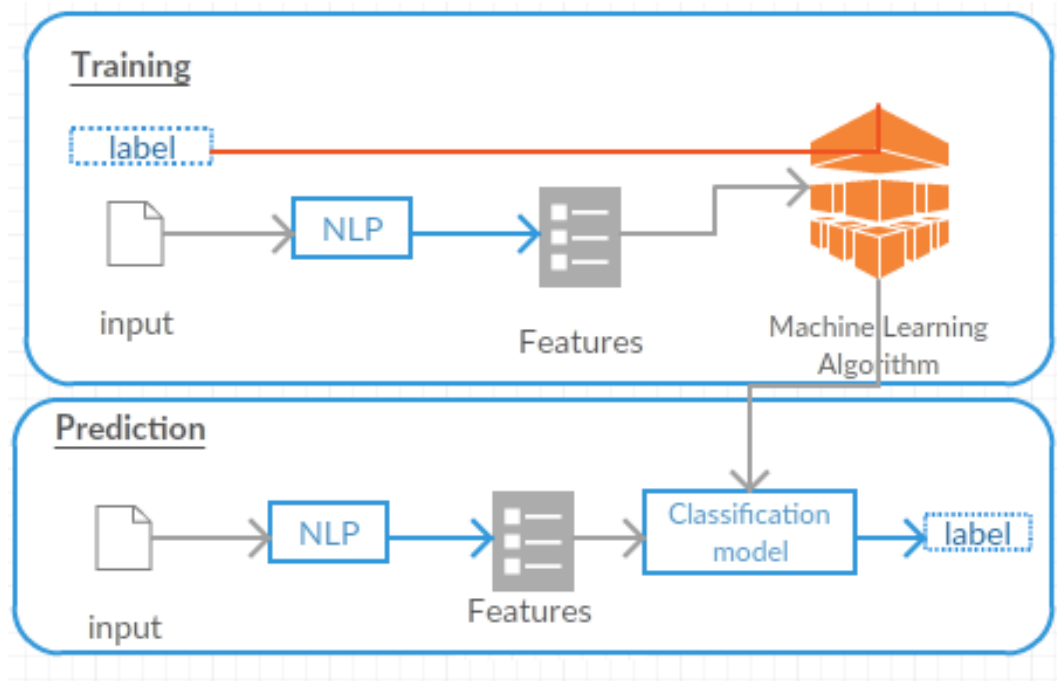


Figure 2.4: Process for text classification

The process of driving out a high-quality information from text is referred as Text mining. It is an interdisciplinary field that draws on IR, DM, ML,

statistics and computational linguistics. Text categorization or classification is a commonly used function of text mining techniques. Text classification is used to resolve categorization problem in library science, information science, and computer science. The main goal of text mining is to choose a correct class label for a given input, that is provided in the form of text. The most important examples of text categorization are spam detection, word-sense disambiguation, and document or web-page classification. The process followed to classify text is shown in Figure 2.4.

Features extraction can be done by following preprocessing steps, tokenization, lemmatization, POST, NER, dependency relationship and term weighting (either TF or TF/IDF).

## 2.4 Equation Solver

An automatic system that is used to solve linear, quadratic, biquadratic, absolute and radical equations is known as equation solver. There exist many online and offline applications or systems that are used to solve arithmetic and linear equations.

Mostly, equation solver applications convert given equation into matrices and then solve it either by Cramer rule or Matrix Inevitability theorem. Symbolab, Mathway, and CyMath are online sources which are used for solving all mathematics problems.

**Symbolab** Symbolab is an advanced mathematics education tool that allows users to learn, practice and discover math topics using mathematical symbols and scientific notations as well as text. It provides automated step by step solutions for solving calculus, algebra, geometry and statistics. Could solve algebraic equation through two methods, i.e., using substitution method and using elimination method.

**Mathway** Mathway is a tool that helps the students to understand and solve their math problems. Mathway can solve a linear algebraic equation system using various methods that are:

- Solve by Substitution
- Solve by Addition/Elimination
- Solve by Graphing
- Solve Using a Matrix with Cramer's Rule

- Determine if Dependent, Independent, or Inconsistent
- Solve Using Matrices by Row Operations
- Solve Using an Inverse Matrix

**CyMath** Cymath is a privately-held company based in New York that believe on open education, with a focus on the “what”, but more on the “how” and the “why”. It can Solves calculus and algebra problems online with a step by step process. It uses three methods for solving linear algebraic problems that are using substitution, elimination, and Matrices by Row Operations. The most widely recognized libraries in java for parsing and assessing numerical expressions are: Java Expression Parser (JEP), Exp4j, formula4j. These libraries permit to enter the self-assertive equation as a string and immediately evaluate it.

- JEP support user defined functions, variables, Unicode characters, Boolean expressions and matrices. It solves linear equations by applying Cramer rule on the matrices that are derived from these equations.
- Exp4j is an expression evaluator taking into account Dijkstra’s Shunting Yard. It’s openly accessible and redistributable under the Apache License 2.0 yet provide support for linear expressions.
- Formula4j parses its input using a grammar strictly limited to mathematical expressions. It does not prove support for string functions, complex numbers, and matrices.



# Chapter 3

## LITERATURE REVIEW

Over the past decade, many approaches have been proposed that exploit text mining techniques to solve mathematics problems. Algebraic word problems are inherently complex and therefore there exist not a single system for providing solutions to the word problems without any restriction. However, some research work related to this area provides guidelines for solving algebraic word problems. Our work is related to three main areas of research: semantic interpretation, information extraction, and automatic word problem solvers [5].

The focus of this chapter is to present an overview of the existing techniques that have been used for the problems extraction and classification. There are two broad categories of the existing work:

- Only NLP (Natural Language Processing) based approaches
- NLP and ML (Machine Learning) based approaches

These techniques have been discussed in detail in the following sections.

### 3.1 Only NLP based approaches

Approaches that are applied to solve a mathematical word problem without applying any classification technique are categorized as *Only NLP based approaches* [6]. In these approaches, the main goal of researchers is to align a text into underlying entities [8]. This alignment might be done with or without natural language processing. There exist many types of research in NLP field that have been focused on understanding semantics and meaning of natural language. Preferably, the automatic problem solver utilizes the information that can be extracted from the problem itself to

show-off the mathematical relationships amongst the features or attributes. Additionally, automatic solver also utilizes some intelligence to come up with a solution to a given mathematical word problem. Research towards automatic solution area dates back to early 1960s with the development of STUDENT program written by Bobrow (1964) [6]. The STUDENT was the first ever program that capable to read, understand and then solves an algebraic word problem. These word problems were must be represented with a confined set of English language. This program only gives the final answer in the English language for a given word problem. Earlier work, on solving logic and math-word problems, uses manually aligned meaning representations or domain knowledge (i.e., on condition that the semantics of all the words are provided) [13].

WORDPRO proposed by Fletcher (1985) [14] was used for solving and understanding arithmetic word problems. To symbolize the meaning of a word problem, WORDPRO used a set of propositions that are supportive for understanding the meaning of a problem text artificially. Supplementary a progressive simulation program known as ROBUST was developed by Bakman (2007) [15] to perceive free-formatted multi-step arithmetic word problems. ROBUST do fragmentation on a problem text to make sentences that are further parsed into propositions. Those propositions that are having complex change-verbs are splitting into fundamental propositions. At that point, the formula(s) are applied on those fundamental propositions by means of substituting the constant values with respect to their corresponding variables.

As *Only NLP based approaches*, do not use ML techniques on NLP-processed data, they cater each question as a separate unit and solve it according to their semantics. So, the systems, that are developed using this approach, do not include any type of artificial intelligence.

## 3.2 NLP and ML based approaches

The systems, that used these approaches, implement the ML algorithms on NLP-processed data to automatically classify text-based questions [6]. The main goal of the researchers to use this category for building the models of solving any math's and logic word problem. Cetintas, Suleyman, et al (2009) [7] proposed a novel approach for text categorization of mathematical word problems, named as Multiplicative Compare and Equal Group problems in their paper. To preprocess the given problems, this method used stop-word removal and stemming beside part-of-speech tagging. After preprocessing, either a SVM classifier or probabilistic meta classifier with default setting

was applied on a dataset. The empirical results demonstrate that the probabilistic meta classifier further enhances the categorization precision. As the probabilistic meta classifier uses the combined weighted results of two SVM classifiers with the different word problem(s). Bussaba and Suma (2014) [16] designed and constructed a mathematical word problems solver in a systematic way, it is a stage towards empowering intellectual capacity in mathematics and showing critical thinking for youngsters in their basic grades.

The template-based learning model by Kushman et al. (2014) [5] was development to enhance the factor of intelligence. This model makes reasons across sentence boundaries to construct and then solve a system of linear equations. While solving, the model could simultaneously perform recovering an alignment of the variables and numbers in these equations to the problem text. Although this model is efficient for solving algebraic word problems but only relies upon training words and NER of a problem text. Another system proposed by Hosseini et al. (2014) [4] is ARIS which maps the problem text into a state representation that involves a set of entities, their containers, quantities, attributes, and relations to solve simple arithmetic word problems. ARIS first splits the problem text into fragments where every single fragment corresponds to an observation or a redesign of the quantity of an entity that may be contained in one or two containers. The verb in every sentence is connected with a couple of containers, and ARIS needs to arrange every verb in a sentence into one of seven categories that portray the effect of the verb on the containers. Although ARIS gives 76+% accuracy but only efficient for arithmetic word problems. This approach does not provide solution toward solving all arithmetic's (i.e., perform  $+$ ,  $-$ ,  $*$ ,  $/$  operations on an equation) and algebraic (i.e., perform all type operations on more than one equation) word problems. Also running the parser was relatively time-consuming. In contrast, the algorithm proposed by Subhro Roy et al. (2015) [2] does not require any additional annotations and could deal with a more general category of problems. The algorithm uses an arithmetic expression tree that would map the problem text into arithmetic expression.

Lipu Zhou et al. (2015) [17] presents another algorithm to consequently tackle algebraic word problems. The problems solved by means of analyzing the hypothesis space containing all possible equation systems. The possible equations were generated through assigning the constants in a given word problem into a set of equation system templates. The equation system was efficiently evaluated as a quadratic programming (QP) problem. Experimental results demonstrate that Lipu Zhou et al. algorithm accomplishes 79.7% accuracy that is about 10% higher than the Kushman et al algorithm.

### 3.3 Critical Analysis

In the previous sections, we discussed two different approaches to solve math-word problems. The section will critically analyze both techniques by discussing their strengths and limitations.

- *Only NLP based approaches* do not use classification yet just use NLP while solving text problems. For instance, these non-ML approaches parse each question to extract equations directly from the problem text, so they require much time and storage. Even though they drive equations directly from the given text yet unable to give high accuracy and precision. However, the non-ML approaches do not require storage, training and compiling time.
- *NLP and ML based approaches* used different classification techniques to inherit artificial intelligence besides NLP. These approaches require enough storage and compiling time while performing both parsing and modeling. The strength of these approaches is their accuracy as they do not require to extract equations from the problem text instead a prediction has been occurred based on the classifier.

*In a nutshell, we can conclude that* there has been researches on automatically solving various types of mathematical word problems. Almost all existing research carried on that topic only have to focus on word count, POS or NER tagging of a problem. As dependency and co-reference parsing gives the co-reference and dependency relationship amongst attributes, so evaluation can give more accurate result if dependency and co-reference parser used beside with POS and NER tagging. Also, the existing techniques require a large number of resources in a form of storage and memory.

### Summary

In this chapter, an overview of existing approaches for solving words problems of mathematics have been presented. The two approaches that have been discussed are *Only NLP based approaches*, and *NLP and ML based approaches*. An overview of these approaches with respect to their working methodology has been discussed. Also, a critical analysis of both approaches is presented.

## Chapter 4

# PROPOSED SYSTEM DESIGN

This chapter provides detail about the architecture and modules used for the realization of our proposed system. The system consists of three main modules.

- Natural Language Processing (NLP)
- Equation Prediction
- Equation Solving

Each module will be discussed in detail in sections. The system was developed by using Stanford NLP, Jazzy, Weka and JEP (Java Expression Parser) that are Java based libraries. We used Eclipse IDE for developing web-based application named as math-word. Figure 4.1 shows the architecture of the proposed system. A corpus of algebraic and arithmetic questions is given as an input to the system. Each question has been pre-processed before applying NLP tools. A classification algorithm is a step toward predicting the equation template which is applied after recognizing dependency relationship amongst entities. At the end, extracted equations would be solved through an API named as JEP.

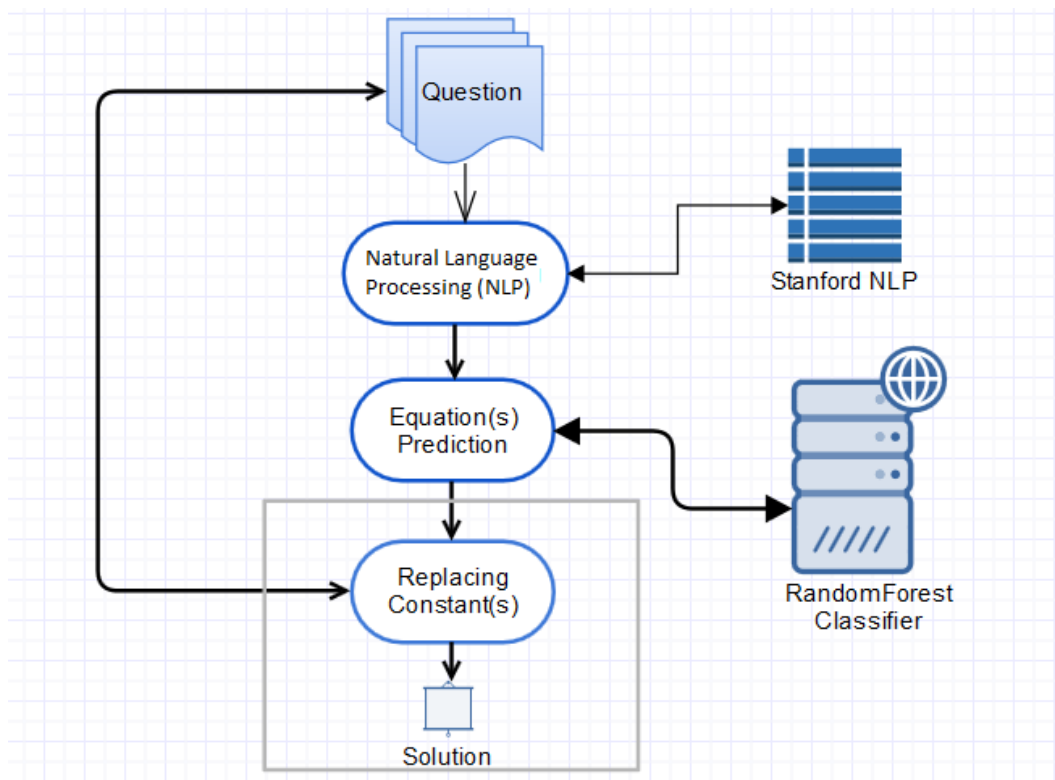


Figure 4.1: System Architecture

## 4.1 Natural Language Processing (NLP) Module

The first module of the proposed methodology is to process the dataset using NLP tools. NLP is for understanding the structure and application of language. Useful tools for getting NLP preprocessed text data are Lemmatization, Stemming, Stop-word Removal, Spell-checking Algorithm, Part of Speech (POS), Named Entity Recognition (NER) and Dependency Parsing (Figure 4.2).

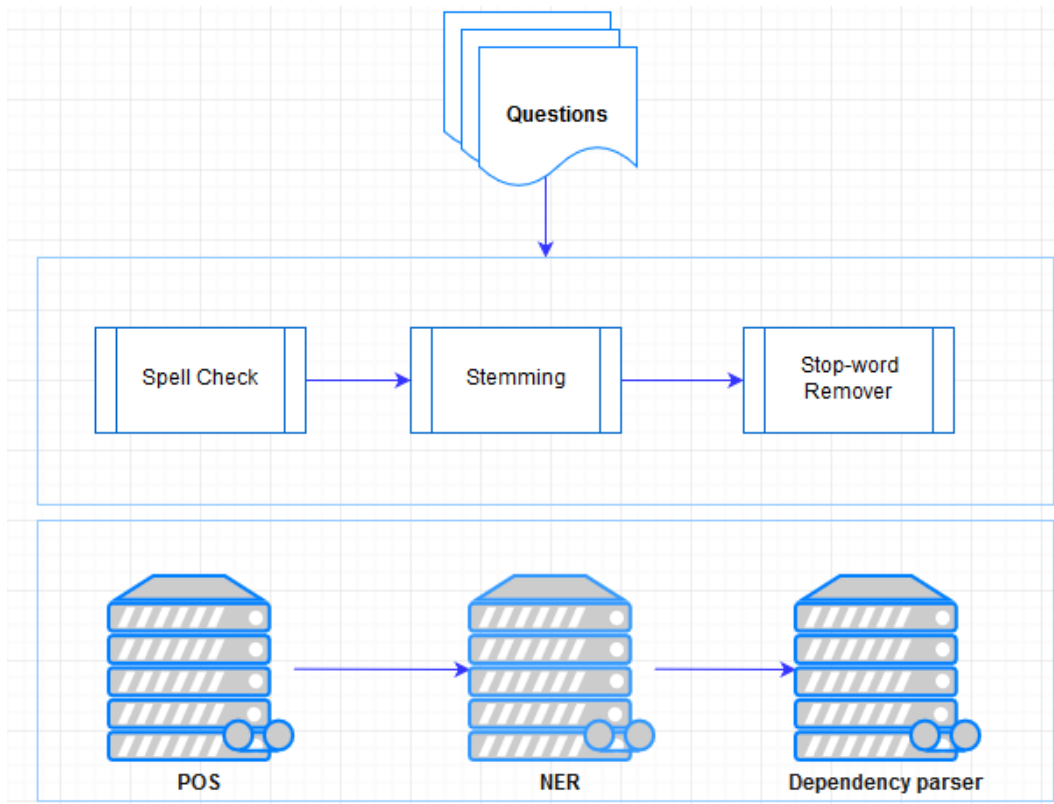


Figure 4.2: Pre-processing Module

Lemmatization is a process of grouping varied forms of a word that can be analyzed and processed as a single item. Whereas stop-words are those words that can be filtered out before actual processing of text data. Lemmatization and Stop-words has mostly done by using rules-based approaches. The basic purpose of stemming is to eliminate several suffixes, to shrink the size of a sentence and to save memory space. Universally used algorithm for stemming is M.F. Porters Algorithm [18]. The main difference between lemmatization and stemming is that a stemmer operates on a single word without know-how of its context. Spell-check utility is an efficient process that was constructed using an efficient string pattern checking algorithm named as Boyer Moore's Algorithm [19]. By using Boyer's model spell-check can skip grammatically incorrect word from pre-processing and entity recognition modules. In the proposed system, each question is checked grammatically through spell checker API named Jazzy that would also give all possible suggestion toward incorrect words. As spell check and stop-word remover are performed before applying other NLP tools, so extracted features do not involve any wrong named entities. Features extraction is possible through three critical NLP

tools that are: POS, NER and Dependency Parser. Following steps would be followed in sequence to extract features from a dataset:

- Tokens are tagged through Part of Speech (POS) tagger.
- Each POS tag is recognized by Named Entity (NE) recognizer.
- Dependency relations are found between word pairs by using dependency parsing.

Each sub-modules of NLP module would be discussed further in the following sub-sections. We used Stanford CoreNLP which is a java based NLP tool that provides a practical approach for language processing. It is an integrated framework that is used for applying a bunch of linguistic analysis tools to a piece of text. CoreNLP integrates many of Stanford's NLP tools, including the POS tagger, the NE recognizer, the parser and the co-reference resolution system.

#### 4.1.1 Spell Check

Before applying any other preprocessing or NLP technique, a given question is checked against its grammatical structure. It is a tool that finds common spelling errors and grammatical mistakes in any language. There exist numerous valuable tools for looking at the grammatical structure of a document alongside with giving all possible suggestions. Some of java based APIs for checking and correcting spells mistakes are:

- JSpell is a checker SDK for the Java J2EE Platform, it has a 6-month trial.
- Google's Spell Checker API v2 is easy to use, yet not available as offline.
- LanguageTool is an open source API that provides an offline solution.
- Jazzy is also a java based API, provides all possible suggestions alongside with identifying the incorrect grammatical structure of a sentence.

Jazzy has been used as a spell checker in our application. The process of checking spell is:

- Scan the text and extract the words contained in it.
- Compare each word with a given dictionary that is a list of correctly spelled words.



- Put misspellings word with all possible suggestions into a list.

Applying Spell check as a preprocessing step would improve the accuracy of text mining techniques. As it avoids wrong matches that will save time, i.e., the matching of misspelled words.

### 4.1.2 Stemming

Stemming is a procedure of NLP in which a word is stemmed to its root. For instance, *giving, given, and got* has a stem word *give*. The motivation behind this strategy is to remove various suffixes that would reduce the number of words, save memory space and time. Stemming is performed after word tokenization of a sentence. Tokenization is about to break a sentence into tokens, e.g., words, phrases, symbols or meaningful elements. These tokens are further processed by other NLP tools like NE recognition, POS tagging, and dependency parsing.

Stemming improves accuracy as it reduces the number of attributes used for a classification algorithm. In the proposed algorithm, we used a morphological analyzer that is used for identifying stems of a word. It takes tokens and their part-of-speech tags as input and generates affix against each word. A morphological analyzer simplifies the task of relation recognition as it groups all the morphologically related forms of a word down to a single root value. Table 4.1 shows the root values generated from tokens of the following example sentence by a morphological analyzer.

**Example 3** *A bank teller has 54 5-dollar and 20-dollar bills in Bank. The value of the bills is 780 dollars. How many 5 dollars' bills are there?*

Table 4.1: Root identification from tokens

Token	Root
has	be
bills	bill
dollars	dollar
The	the
is	is
How	how
are	is

For our algorithm, we require affix for numeric (num), prepositional (prep), possession (poss), adjectival (amod) and adverbial (advmod) modifiers that are dependency relations.

### 4.1.3 Stop-word Remover

Stop words are those words that have been used more frequently in a language. They are language-specific functional words which carry no information. Such as pronouns, prepositions, conjunctions. These words are pointless in Text mining, so eliminating these words from the training data would be used in sense of accuracy and precision. In our algorithm, all NER objects (O) and special characters such as semicolon, colon and percentage sign are stop-words. We used a rule-based remover algorithm to eliminate stop words in our dataset. By removing the stop words, we got dimensionally a reduced dataset.

### 4.1.4 POS Tagger (POST)

Identification of a part-of-speech for a given word is an important precursor to any information extraction task. POS tagger is a tool used to distinguish parts of speech for each word. Before applying POST, text has to be segmented into linguistically significant units, such as words, symbols, numbers, and punctuation. This process is referred as tokenization. These words then be tagged with different Part Of Speech (POS) such as verb, noun, adjective, adverb etc. POS information for each token is required by the relation extraction modules. Table 4.2 shows the output for POS tagging of the example 3 sentence. The tokens of the sentence are listed in the first column, whereas the corresponding tags in the second and their descriptions in the third column.

### 4.1.5 Named Entity Recognition (NER)

We used seven class model for entity recognition to recognize entities in each individual question. This phase identifies name of entities that are the names of location, person, organization, money, percent, date and time. If a token does not fall in any of the above class, it is categorized as a general object (O).

POS tag is a pre requirement for NE recognitions. Table 4.3 shows the entity relations generated from POS tags and the NER model for the example 3.

Table 4.2: POS tags for example 3

Token	Tag	Description
teller	NN	Personal Noun
has	VBZ	Verb
55	CD	Cardinal number
5-dollar	JJ	Adjective
and	CC	Conjunction
bills	NNS	Noun
In	IN	Preposition
Bank	NNP	Personal Noun
The	DT	Determiner
value	NN	Noun
Of	IN	Preposition
is	VBZ	Verb

Table 4.3: NER for example 3

Token	NE relation
teller	O
has	O
54	NUMBER
5-dollar	NUMBER
bills	O
Bank	ORGANIZATION
780	MONEY
dollars	MONEY

### 4.1.6 Dependency Parsing

This module uses the result from POST and apply dependency parsing that gives the exact relationship between two consecutive words. CoreNLP uses a *Neural Network Dependency Parser* for finding out the dependency relations. The Neural Network Dependency Parser builds parse by performing a linear-time scan over the words. At every step, it maintains a stack of words which are currently being processed, and a buffer of words yet to be processed. The initial state is to have all of the words in order on the buffer, with a single ROOT node on the stack. The following transitions could be applied:

1. *LEFT-ARC*: marks the second item on the stack as a dependent of the

first item, and removes the second item from the stack.

2. *RIGHT-ARC*: marks the first item on the stack as a dependent of the second item, and removes the first item from the stack.
3. *SHIFT*: removes a word from the buffer and pushes it onto the stack.

These transitions are chosen using the neural network classifier in each state. Figure 4.3 shows full algorithm for Neural Network Dependency parser.

<b>Configuration:</b>	$(S, B, A)$	$[S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}]$
<b>Initial:</b>	$([], [0, 1, \dots, n], \{\})$	
<b>Terminal:</b>	$(S, [], A)$	
<b>Shift:</b>	$(S, i B, A) \Rightarrow (S i, B, A)$	
<b>Reduce:</b>	$(S i, B, A) \Rightarrow (S, B, A)$	$h(i, A)$
<b>Right-Arc(<math>k</math>):</b>	$(S i, j B, A) \Rightarrow (S i j, B, A \cup \{(i, j, k)\})$	
<b>Left-Arc(<math>k</math>):</b>	$(S i, j B, A) \Rightarrow (S, j B, A \cup \{(j, i, k)\})$	$\neg h(i, A) \wedge i \neq 0$
<b>Notation:</b>	$S i = \text{stack with top } i \text{ and remainder } S$ $j B = \text{buffer with head } j \text{ and remainder } B$ $h(i, A) = i \text{ has a head in } A$	

Figure 4.3: Algorithm for Neural Network Dependency Parser [2]

Table 4.4 shows the dependency relations for a sentence “A bank teller has 54 5-dollar and 20-dollar bills in her cash drawer.” from example 3.

## 4.2 Equation Prediction Module

We have ready to use a dataset after finding out POS tags, Entity relation and dependency relation. The *Equation Prediction Module* is for training and predicting equation template against each question (Figure 4.4). To get the equation template, this module follows three steps that are named as: TF calculation, generation of similarity matrix and classification. Term frequency is calculated against each document. The document consists of POS tags, entity relation, dependency type and some of the dependency relations that are prep, num, poss, amod and advmod. A similarity matrix is generated for applying a classification algorithm on it [3]. In the following

Table 4.4: Dependency relations for a sample sentence

Word pair (governor, dependent)	Dependency type	Description
ROOT, has	root	
teller, A	det	Determiner
teller, bank	nn	Noun compound modifier
has, teller	nsubj	Nominal subject
5-dollar, 54	num	Numeric modifier
has, 5-dollar	dobj	Direct object
5-dollar, and	cc	Coordination
bills, 20-dollar	amod	Adjective modifier
5-dollar, bills	conj	Conjunct
5-dollar, in	prep	Prepositional modifier
drawer, her	poss	Possession modifier
drawer, cash	nn	Noun compound modifier
in, drawer	pobj	Prepositional object

subsections, we will discuss detailed steps followed by equation prediction module.

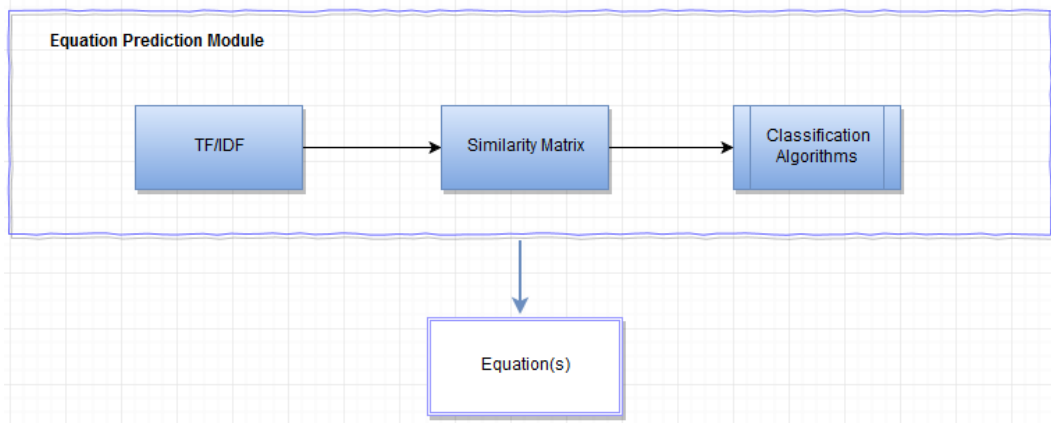


Figure 4.4: Equation Prediction Module

### 4.2.1 Term Frequency and Similarity Matrix

Term Frequency (TF) measures the count of term  $i$  that occur in a document  $j$ . TF is useful in the field of natural language processing as it provides aids to ease text mining using data mining algorithms. For calculating TF of a  $i$

word in  $j$  document, a formula is:

$$tf_{ij} = N_{ij}/N_j \quad (4.1)$$

where  $N_{ij}$  is a count of a word  $i$  in  $j$  document and  $N_j$  is total number of count in  $j$  document.

There exist two most common methods in java for calculating TF, both are equally useful, that are:

### Using Eclipse Collections method

It converts the list of words to a Bag by keeping multiplicity, i.e.,

```
Bag < String > words = Lists("bye", "bye", "ciao", "bye", "ciao").toBag()
```

The occurrences (count) of a bag word can be easily found, i.e.,

```
countCiao = words.occurrencesOf("ciao")
```

### Using map-and-reduce method

This method converts a list of words into a stream of words, and then group them according to their unique identity, i.e.,

```
Map < String, int > collect = wordsList.stream().
```

```
collect(groupingBy(Function.identity(), counting()))
```

TF of individual words is not useful until it is presented in the form of a vector that could be achieved using a similarity matrix.

A similarity matrix or document-term matrix is a two dimensional matrix which contains TF against each document. Where rows correspond to documents in a corpus and columns correspond to terms, so each entry  $(i, j)$  represents term frequency. For instance, if a corpus has the following three documents:

Document1: "Subtract two from three"

Document2: "Add two and three"

Document3: "Subtract three from one"

Table 4.5 shows the similarity matrix in correspondence to TF.

Table 4.5: Similarity Matrix on TF

	<b>ADD</b>	<b>AND</b>	<b>FROM</b>	<b>ONE</b>	<b>SUBTRACT</b>	<b>THREE</b>	<b>TWO</b>
<b>DOCUMENT1</b>	0	0	1	0	1	1	1
<b>DOCUMENT2</b>	1	1	0	0	0	1	1
<b>DOCUMENT3</b>	0	0	1	1	1	1	0

### 4.2.2 Classifier Algorithm

In this module, we used a Random Forest (RF) classification algorithm that is applied on a similarity matrix of our dataset. The algorithm gives us equation(s) that would be solved through the equation solving module. RF algorithm constructs a forest of random trees with attributes chosen at random, then uses an ensemble of decision trees by a bootstrap sample of the training set (Figure 4.5).

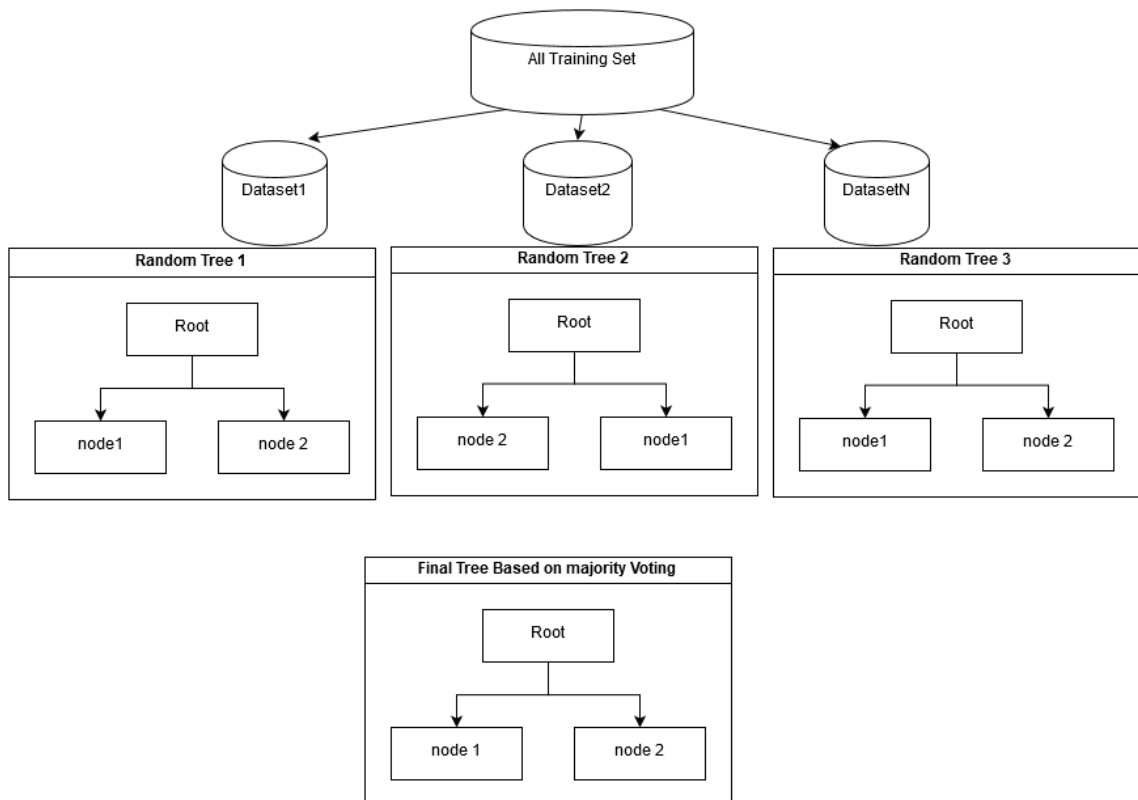


Figure 4.5: Random Forest Classifier

In Weka, the default parameters set for RF Classification are:

–*I* <number of trees> 100

–*K* <number of features> considered while creating each random tree, by default value is 0 that mean not specific.

–*depth* <num> the maximum depth of the trees, by default it is 0 that is for unlimited.

Weka generates an RF on hundred randomly generated trees, where each tree created through random attributes. An ensemble chooses one tree out of hundred based on majority voting. In 10-fold cross validation method, ten randomly selected trees are used to generate 10 RF tree iteratively. The final tree is selected out of 10 RF tree based on majority voting.

We chose RF for our proposed system as it could have performed better on datasets having multiple classes. Also, gives more accurate result than SVM (Support Vector Machine) and KStar Algorithm as RF is used 100 trees, each constructed while considering 9 random features.

The predicted equation for example 3 is “ $b * x + c * y = d; 1 * x + 1 * y = a$ ”.

### 4.3 Equation Solving Module

This module converts an equation template into an equation system against given problem and provides an evaluated result of the converted equation(s). Figure 4.6 shows the process of conversion and evaluation of an equation system.

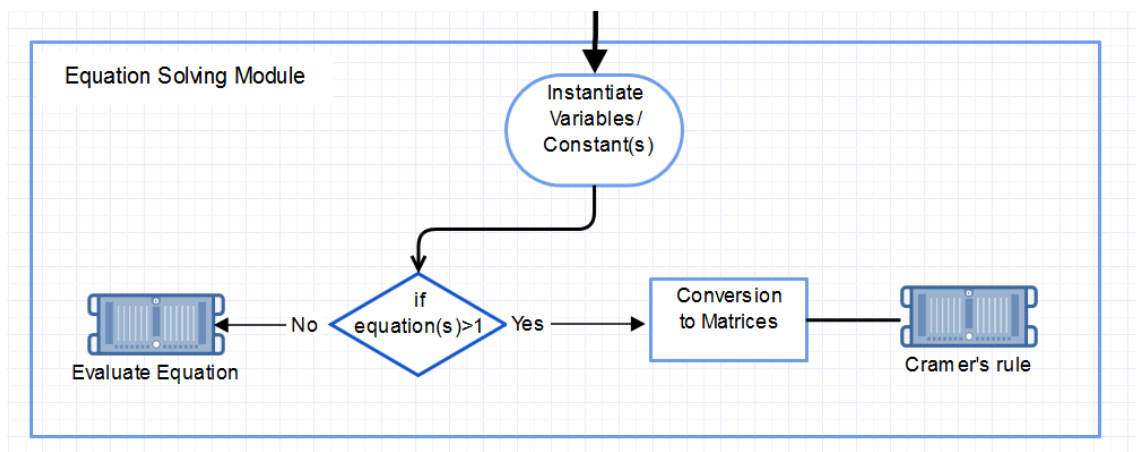


Figure 4.6: Equation Solving Process

An equation template that is selected through a classification algorithm is instantiate with variables and constants for a given word problem. If the



instantiated equation only contains one variable it is evaluated instantly using *Evaluate Function* of JEP (Java Expression Parser), otherwise equations are converted into a matrix system, and *Cramer Rule* of JEP is applied to solve it.

The solution of example 3 is “ $x = 20, y = 34$ ”.

## Summary

In this chapter a detailed discussion of the proposed methodology has been presented. The three main modules of the system, i.e. NLP module, equation prediction module and equation solving module have been described in detail.

# Chapter 5

## IMPLEMENTATION AND EVALUATION

This chapter has two main parts. The first part will discuss the technical details about the system implementation and the second part will focus on the evaluation of the proposed system.

### 5.1 System Implementation

In this section we will discuss system specification, software specification, and output of the proposed system, that are illustrated through a series of screen shots.

The system specifications, used in the development of the system, are shown in Table 5.1.

Table 5.1: System Specification

Processor	Intel 1.7GHz Corei3 4010U 32-bit or 64-bit
RAM	4 GB
Operating System	Windows 8.1 Pro
Hard disk space	16 GB

Table 5.2: Software Specification

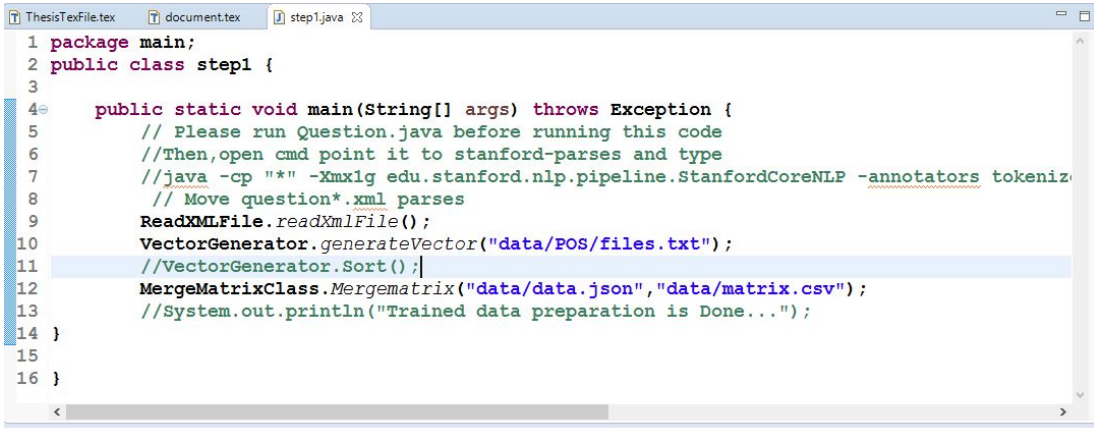
Development Language	Java version 8 update 40
Designing Language	HTML, CSS, jQuery
IDE	Eclipse & Weka
Libraries	StanfordCoreNLP, JEP & Weka

The software specifications, used in the development of the system, are shown in Table 5.2. We used Eclipse IDE (Integrated Development Environment) to develop our proposed system as it is free of cost and have a lot of useful built-in libraries.

As discussed in chapter 2, the process for text classification involves two steps training and prediction (Figure 2.4). The training step is further sub divided into two sub-steps that are: (i) preprocessing and NLP processing dataset (ii) applying machine learning algorithm.

Figure 5.1 shows the sub-code to get a processed training set from a raw dataset. Getting a trained dataset involves five steps:

1. Put each question into a separate document from a given corpus of dataset that is having 752 questions.
2. Find annotator against each document by using StanfordCoreNLP library that will generate XML file containing annotators.
3. Read XML files and find out only useful POS tags, NER and Dependency parsing relation.
4. Generate a similarity matrix contains term frequency against each word in a document
5. Merge similarity matrix with equation templates that are already extracted in dataset.



```

1 package main;
2 public class step1 {
3
4     public static void main(String[] args) throws Exception {
5         // Please run Question.java before running this code
6         //Then, open cmd point it to stanford-parses and type
7         //java -cp "*" -Xmx1g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokeniz
8         // Move question*.xml parses
9         ReadXMLFile.readXmlFile();
10        VectorGenerator.generateVector("data/POS/files.txt");
11        //VectorGenerator.Sort();|
12        MergeMatrixClass.MergeMatrix("data/data.json", "data/matrix.csv");
13        //System.out.println("Trained data preparation is Done...");
14    }
15
16 }

```

Figure 5.1: Training dataset implementation

The second step involves in training is using a machine learning algorithm on a given dataset. Figure 5.2 shows, RandomForest classifier is applied on the dataset.



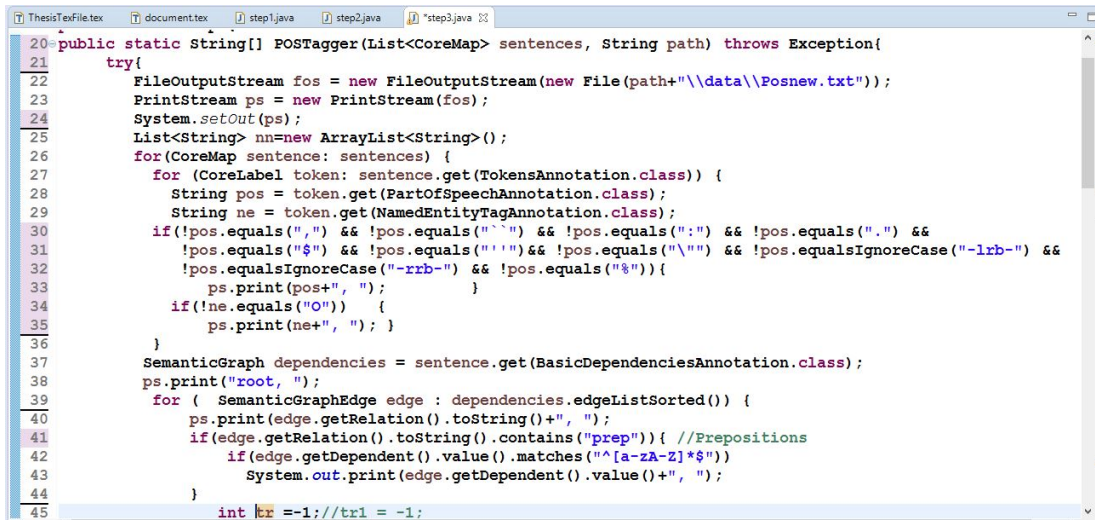
```

1 package main;
2 import weka.core.converters.ConverterUtils.DataSource;
5 //Run the code to generate matrix for new example
6 public class step2 {
7     static void train() throws Exception{
8         DeleteFiles.deleten();
9
10        DataSource source = new DataSource("data/trainingData.csv");
11        Instances data = source.getDataSet();
12        data.setClassIndex(data.numAttributes() - 1);
13        RandomForest cls = new RandomForest();
14        cls.buildClassifier(data); //Training
15        weka.core.SerializationHelper.write("data/classifier.model", cls);
16    }
17
18    public static void main(String[] args) throws Exception {
19
20
21        train();
22    }
23 }

```

Figure 5.2: Training through RF classifier

Figure 5.3 shows the prediction step and Figure 5.4 shows web-based interface of proposed system.



```

20 public static String[] POSTagger(List<CoreMap> sentences, String path) throws Exception{
21     try{
22         FileOutputStream fos = new FileOutputStream(new File(path+"\\data\\Posnew.txt"));
23         PrintStream ps = new PrintStream(fos);
24         System.setOut(ps);
25         List<String> nn=new ArrayList<String>();
26         for(CoreMap sentence: sentences) {
27             for (CoreLabel token: sentence.get(TokensAnnotation.class)) {
28                 String pos = token.get(PartOfSpeechAnnotation.class);
29                 String ne = token.get(NamedEntityTagAnnotation.class);
30                 if(!pos.equals(",") && !pos.equals("`") && !pos.equals(":") && !pos.equals(".") &&
31                     !pos.equals("$") && !pos.equals("'") && !pos.equals("\"") && !pos.equalsIgnoreCase("-lrb-") &&
32                     !pos.equalsIgnoreCase("-rrb-") && !pos.equals("%")){
33                     ps.print(pos+" ");
34                 }
35                 if(!ne.equals("O")) {
36                     ps.print(ne+" ");
37                 }
38             }
39             SemanticGraph dependencies = sentence.get(BasicDependenciesAnnotation.class);
40             ps.print("root, ");
41             for ( SemanticGraphEdge edge : dependencies.edgeListSorted()) {
42                 ps.print(edge.getRelation().toString()+" ");
43                 if(edge.getRelation().toString().contains("prep")){ //Prepositions
44                     if(edge.getDependent().value().matches("[a-zA-Z]*"))
45                         System.out.print(edge.getDependent().value()+" ");
46                 }
47             }
48             int tr = -1; //tr1 = -1;

```

Figure 5.3: Predicting through RF classifier

**Solution for Question**

' A bank teller has 54 5-dollar and 20-dollar bills in her cash drawer. The value of the bills is 780 dollars. How many 5 dollars bills are there?'

---

Equation(s):  $b*x+c*y=d;1*x+1*y=a$   
 So,

Converting x into --> x  
 After adjusting variables in equation(s) :  
 $b*x+c*y=d,$   
 $1*x+1*y=a,$   
 Final equation(s) will be:  
 Equation 1 :  $5*x+20*y=780,$   
 Equation 2 :  $1*x+1*y=54.0,$   
 Equation(s) solved through matrix, So matrix of the given equation(s)  
 $|5.0\ 20.0| |x| = |780.0|$   
 $|1.0\ 1.0| |y| = |54.0|$   
**Result:**  
 $x= 20.0 --> 20,$   
 $y= 34.0 --> 34$

Figure 5.4: Web-Based Interface

## 5.2 Dataset Specifications

In our experiment, we used a dataset from two different sources to evaluate and compare our system with existing math-word problem solvers. The specification of each data source is described in the following sub-sections. An equation template against each question is manually prepared by concerting experts. A sample of our dataset is shown in Figure 5.5.

### 5.2.1 Online Source

We collected a live dataset of algebra word problems from algebra.com [20]. As questions are posted by students, so these problems are highly varied and taken from real problems faced by students. We randomly chose 500 linear algebra questions which did not require any explicit background knowledge.

### 5.2.2 Book Source

We collectively got 252 questions from the following syllabus books:

A writing workshop enrolls novelists and poets in a ratio of 5 to 3. There are 24 people at the workshop. How many novelists are there? How many poets are there?	$b*x-a*y=0;1*x+1*y=c$
You are selling tickets for a high school play. Student tickets cost 4 dollars and general admission tickets cost 6 dollars. You sell 525 tickets and collect 2876 dollars. How many student tickets and general admission tickets did you sell?	$a*x+b*y=d;1*x+1*y=c$
A store is selling compact discs for 10.50 dollars and 8.50 dollars. You buy 10 discs and spend a total of 93 dollars. How many compact discs did you buy that cost 10.50 dollars? How many did you buy that cost 8.50 dollars?	$a*x+b*y=d;1*x+1*y=c$
Bob invested 6100 dollars in 2 funds , which pay 7 % and 6 % simple interest respectively. The combined interest he earned for both funds was 405 dollars for one year. How many dollars was invested at 7 %? How much was invested at 6 %?	$b*x+c*y=d;1*x+1*y=a$
Your teacher is giving you a test worth 100 points and containing 40 questions. There are 2 point and 4 point questions on the test. How many two point questions are there? How many 4 point questions?	$c*x+d*y=a;1*x+1*y=b$
You have 160 dollars and save 7 dollars per week. Your friend has 210 dollars and saves 5 dollars per week. After how many weeks will each of you have saved the same amount of money?	$x=(a-c)/(d-b)$
A theater has 80 seats. On opening night , they sold out , selling adult tickets for 12 dollars each and child tickets for 5 dollars each. If they made a total of 519 dollars , how many child tickets were sold?	$b*x+c*y=d;1*x+1*y=a$

Figure 5.5: Sample of the Dataset

- 70 questions from Mathematics-9 [21]; Chapter 4: Algebraic Expressions and Algebraic Formulas, Chapter 6: Algebraic Manipulation & Chapter 7: Linear Equation
- 58 questions from Cambridge o-level mathematics volume 1 [22]; Chapter 3: Beginning Algebra & Chapter 5: Working with Algebra
- 49 question Cambridge o-level mathematics volume 2 [23]; Chapter 2: Algebra I & Chapter 4: Algebra II
- 44 questions from Mathematics-8 [24].
- 31 questions from Mathematics-7 [25].

### 5.3 Evaluation Metrics

We have selected the widely used measures to evaluate the performance of our proposed system that are (i) Recall, (ii) False Positive Rate, (iii) Precision and (iv) F-measure. Before interpreting these terms, it is important to understand TP (True Positive), TN (True Negative), FP (False Positive) and FN (False Negative) (as shown in Table 5.3).

Table 5.3: Confusion matrix [3]

	<b>Prediction: Positive</b>	<b>Prediction: Negative</b>	
<b>Reality: Positive</b>	TP	FN	P
<b>Reality: Negative</b>	FP	TN	N

- *TP*: Both reality and prediction of the model are positive
- *FN*: Reality is positive but prediction of the model goes to negative
- *FP*: Reality has been negative but prediction goes for positive
- *TN*: Both reality and prediction are negative

**Recall** Recall also known as Sensitivity or True Positive (TP) rate, is about how accurately each question of a dataset is classified into their relevant equation template. In other words, it measures how many of the questions that should have been identified are actually identified. Equation 5.1 shows the formula:

$$Recall = \frac{TP}{P} \quad (5.1)$$

**False Positive (FP) Rate** It measures how many of the questions are wrongly classified. In other words, the FP Rate is calculated as the ratio between the number of questions that should have not been identified are actually identified and the total number of not identified questions. Equation 5.2 shows the formula:

$$FP - Rate = \frac{FP}{N} \quad (5.2)$$

**Precision** Precision measures the number of correctly identified questions as a percentage of number of questions identified. In other words, it measures how many of the questions that the system identified are correct. Equation 5.3 shows the formula:

$$Precision = \frac{TP}{(TP + FP)} \quad (5.3)$$

**F-measure** F-measure is the harmonic mean of precision and recall, calculated as:

$$F - measure = 2 * \frac{Precision.Recall}{Precision + Recall} \quad (5.4)$$

## 5.4 Performance Evaluation

We evaluated the proposed system against Hossemi et al. [4] and Kushman et al. [5] systems, and their related datasets. We evaluated the systems to compute Recall, Precision and F-measure.

Table 5.4 shows the evaluation metrics against proposed's dataset. While Table 5.5 and Table 5.6 show the evaluation metrics against Hossemi et al. [4]'s dataset and Kushman et al. [5]'s dataset respectively.

Table 5.4: Evaluation metrics using proposed's dataset

<b>Approaches</b>	<b>Recall</b>	<b>Precision</b>	<b>F-Measure</b>
Hossemi et al. [4]'s	0.80	0.772	0.778
Kushman et al. [5]'s	0.72	0.691	0.697
Proposed System	0.835	0.806	0.812

Table 5.5: Evaluation metrics using Hossemi et al. [4]'s dataset

<b>Approaches</b>	<b>Recall</b>	<b>Precision</b>	<b>F-Measure</b>
Hossemi et al. [4]'s	0.776	0.748	0.753
Proposed System	0.658	0.629	0.64

Table 5.6: Evaluation metrics using Kushman et al. [5]'s dataset

<b>Approaches</b>	<b>Recall</b>	<b>Precision</b>	<b>F-Measure</b>
Kushman et al. [5]'s	0.69	0.671	0.68
Proposed System	0.786	0.766	0.77

The illustrations shown in Table 5.4 and 5.6 demonstrate that the proposed system achieves better results than the template-based method of Kushman et al. Our system improves both precision and recall at the same time. The reason behind the better performance is that our approach uses just POS tags, NER and dependency relations instead of using lemmas against each word that would only increase the number of features. Moreover, the proposed system effectively instantiate nouns and numbers into classified equation templates.

While, Table 5.5 illustrated that the system proposed by Hossemi et al. [4] gives better result, the reason behind is that Hossemi et al. [4] used verb categorization as it is specific for only subtraction and addition problems.



## Summary

In this chapter, we discussed our system's implementation and evaluation methodology. The hardware and software specification have been described. The output of the system modules was illustrated with screen shots. The description of the dataset used for the system evaluation and comparison have been described. The system evaluation against the existing methodologies has been included, also a discussion on system's comparison with existing techniques based on their implementation and evaluation methodologies is presented.

# Chapter 6

## CONCLUSION AND FUTURE DIRECTION

In this chapter, we present a summary of the contributions of the research work documented in this thesis. Some of the fundamental limitations of our approach and an outlook of the future directions where this work can be extended are presented at the end of this chapter.

### 6.1 Conclusion

In this research, a system has been proposed that automatically extract and solve arithmetic and linear algebra word problems. Equations are extracted through template-based method that used preprocessing, NLP and classification algorithm. While these problems are solved through equation evaluation API named JEP (Java Expression Parser). The proposed system targeting students of grade 4 to 9. The proposed system has been evaluated against two existing datasets besides proposed's dataset. The existing datasets have been fetched from Hosseini et al. [4] and Kushman et al. [5]. The system has been evaluated through the proposed and existing system's dataset. The model gives 83+% accuracy. High accuracy is because of using dependency relation besides with POS (Part Of Speech), NER (Named Entity Recognition).

The following sections demonstrate the proposed system's contributions, limitations and future work.

## 6.2 Contributions

**Improved Accuracy** The proposed model got much higher accuracy in comparison of existing template-based algorithms. Improvement in accuracy is the cause of using dependency types and relations.

**Targeting Real Problems** The dataset was collected from two different data sources that are from course books and online portal.

**Simplicity** Compared to other approaches, the proposed approach is fairly straightforward to implement and achieves competitive performance. It is because the proposed model does not use unnecessary features. So it also saves time for learning a model. Additionally, much less computing resources used in comparing with other template-based algorithms.

## 6.3 System Limitations & Future Work

- Does not give throughput toward questions with irrelevant information.
- To further improve the accuracy we intended to incorporate the co-reference resolution technique along with dependency relation during natural language processing.
- This work could be extended to the general domain, i.e., for all mathematical word problems such as geometry, quadratic algebra, statistics, logical and analytical problems.
- Advanced dataset might be used that have also been targeted to higher level problems.

# Bibliography

- [1] G. Hardy and E. Wright, *An introduction to the theory of numbers, 6th edn, revised by DR Heath-Brown and JH Silverman*, vol. 12. Oxford University Press, Oxford, 2008.
- [2] J. Nivre, *Recent Advances in Dependency Parsing*. EACL, 2014.
- [3] X. Wang, J. Tang, and H. Liu, “Document clustering via matrix representation,” in *2011 IEEE 11th International Conference on Data Mining*, pp. 804–813, IEEE, 2011.
- [4] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, “Learning to solve arithmetic word problems with verb categorization,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 523–533, 2014.
- [5] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi, “Learning to automatically solve algebra word problems,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pp. 271–281, 2014.
- [6] A. Mukherjee and U. Garain, “A review of methods for automatic understanding of natural language mathematical problems,” *Artif. Intell. Rev.*, vol. 29, no. 2, pp. 93–122, 2008.
- [7] S. Cetintas, L. Si, Y. P. Xin, D. Zhang, and J. Y. Park, “Automatic text categorization of mathematical word problems,” in *Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference, May 19-21, 2009, Sanibel Island, Florida, USA*, 2009.

- [8] M. Miyani, S. Doshi, and J. Jain, “Word problem solver system using artificial intelligence,” *Procedia Computer Science*, vol. 45, pp. 800–807, 2015.
- [9] G. A. Ascoli, “The mind-brain relationship as a mathematical problem,” *ISRN neuroscience*, vol. 2013, 2013.
- [10] *Pure and applied math.* <http://www.tinyepiphany.com/2012/03/pure-and-applied-math.htm> ed., 2012.
- [11] H. S. M. Coxeter, *Introduction to Geometry*. second ed., 1969.
- [12] B. Qadir, *Generation of Domain Ontologies from Text*. PhD thesis, SEECS-NUST, 2015.
- [13] J. P. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” in *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pp. 384–394, 2010.
- [14] C. R. Fletcher, “Understanding and solving arithmetic word problems: A computer simulation,” *Behavior Research Methods, Instruments, & Computers*, vol. 17, no. 5, pp. 565–571, 1985.
- [15] Y. Bakman, “Robust understanding of word problems with extraneous information,” *arXiv preprint math/0701393*, 2007.
- [16] B. Amnueypornsakul and S. Bhat, “Machine-guided solution to mathematical word problems,” in *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 28, Cape Panwa Hotel, Phuket, Thailand, December 12-14, 2014*, pp. 111–119, 2014.
- [17] L. Zhou, S. Dai, and L. Chen, “Learn to solve algebra word problems using quadratic programming,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 817–822, 2015.
- [18] C. Ramasubramanian and R. Ramya, “Effective pre-processing activities in text mining using improved porters stemming algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 12, pp. 2278–1021, 2013.

- [19] D. Munková, M. Munk, and M. Vozár, “Data pre-processing evaluation for text mining: Transaction/sequence model,” in *Proceedings of the International Conference on Computational Science, ICCS 2013, Barcelona, Spain, 5-7 June, 2013*, pp. 1198–1207, 2013.
- [20] *Algabric forum*. <http://www.algabra.com> ed., 2012.
- [21] *Mathematics 9*. Punjab Text Books, Lahore, 2014.
- [22] D. A. Simpson, *Cambridge O Level Mathematics Volume 1*. Cambridge University Press, 2016.
- [23] D. A. Simpson, *Cambridge O Level Mathematics Volume 2*. Cambridge University Press, 2016.
- [24] *Mathematics 8*. Punjab Text Books, Lahore, 2014.
- [25] *Mathematics 7*. Punjab Text Books, Lahore, 2014.